5. Edward

Program Name: Edward.java **Test Input File: edward.dat**

Edward has found a board game similar to Go which uses Xs and Os on an 8X8 checker/chess style board. He wants to write a program to simulate the game and needs help with one part of the AI (artificial intelligence). Given a situation at some point in the game, he wants to know if certain pieces have "captured" other pieces.

For an X piece to capture an O piece, it must be adjacent to it, and have another X on the other side of the O, vertically or horizontally, but not diagonally, also adjacent to it.

For example, in the board shown below, the O at position (6,3) is captured by the Xs at positions (5,3) and (7,3). The same is true for the O at position (6,6), captured by the Xs at positions (6,5) and (6,7). However, the two Os in the bottom right corner at positions (1,8) and (2,8) are not captured horizontally since they are on the edge, even though they are surrounded by X pieces. However, a piece that is on the edge can be captured if there are X pieces on either side alongside of it. For example, if position (1,8) was an X piece, the O piece in position (2,8) would be captured.

```
--------
--X-----
--O-XOX-
--X-----
--------
-------X
------XO
------XO
```

What Edward needs to know is when the position of an X position is indicated, what O piece, if any, is it helping to capture? For example, if the X piece at position (7,3) is indicated, the answer would be the position of the O piece it is helping to capture, which would be (6,3). If the X piece at position (1,7) is mentioned, there is no O piece it is helping to capture, so the answer would be NONE. Since this step is just the starting point in the overall AI of the game, Edward will only use data where at most one O is captured by any X piece. Once he perfects that, he'll move on to multiple possibilities, but not now.

Input: An initial value N, indicating N data sets to follow. Each data set consists of two rows, one for the O pieces, and the next for the X pieces. Each row starts with a value Q, followed by Q ordered coordinate pairs (x followed by y) indicating the positions of the pieces. After the two rows of pieces, an integer P indicates P ordered pairs, each on one line, representing the X pieces to query.

Output: The coordinates of the given X piece and of the O piece it is helping to capture, or the phrase "NONE" if there is no capture, exactly as shown and formatted in the examples below. It is guaranteed that each X piece will help capture at most one O piece. There is exactly one space on either side of the arrow for each output. A blank line will follow each complete output.

Sample Input:

```
2
4 1 8 2 8 6 3 6 6
7 1 7 2 7 3 8 5 3 7 3 6 5 6 7
5
7 3
6 5
5 3
3 8
1 7
4 1 1 2 2 2 6 1 7
9 2 1 1 2 3 2 1 5 2 5 2 3 1 6 2 7 1 8
6
3 2
2 1
1 2
1 5
1 8
2 7
```

Sample Output:

```
(7,3) ==> (6,3)
(6,5) ==> (6,6)
(5,3) ==> (6,3)
(3,8) ==> NONE
(1,7) ==> NONE

(3,2) ==> (2,2)
(2,1) ==> (2,2)
(1,2) ==> (2,2)
(1,5) ==> NONE
(1,8) ==> (1,7)
(2,7) ==> (2,6)
```